Khan CPU

Joseph DePalo, Rishi Bidarkota

December 9, 2023

I pledge my honor that I have abided by the Stevens Honor System.

1 General Specifications

- 16-bit, single-cycle datapath CPU
- 4 16-bit general registers (R0 - R3)
- 16-bit instructions

2 Instruction Set Architecture

2.1 Breakdown of an Instruction

2.1.1 Control Signals

5 control signals are used to achieve the desired behavior of each function. Their ARM equivalents are listed in parentheses.

- Bit 0: RegWrite
- Bit 1: UseImm (Reg2Loc, ALUsrc)
- Bit 2: ALUop
  - 0 corresponds to addition
  - 1 corresponds to subtraction
- Bit 3: MemWrite
- Bit 4: LoadMem (MemRead, MemToReg)

2.1.2 Registers

Khan CPU has 4 general registers. As such, we need $\lceil \log_2 4 \rceil = 2$ bits to represent a register.

2.1.3 Putting Things Together

We need 5 bits to represent the opcode for any given instruction and 6 bits to represent 3 registers. This brings us to 11 bits, so it makes most sense to represent an instruction as 16 bits, or the smallest power of 2 greater than 11. For instructions that use 2 registers and an immediate value we need 5 bits for the opcode, 4 bits for the registers, and that leaves 7 bits for the immediate value.

2.2 Instructions

2.2.1 Explaining Instruction Terminology

- Rd denotes the destination register for an arithmetic operation.
- Rt denotes the target register for a memory operation.
- Rn and Rm are operand registers. How they are used are detailed below.
- imm7 is a 7-bit immediate value, or a signed integer in the range of -64 to 63.
- filler5 represents 5 filler bits which will have no impact on the program no matter their value. Our assembler represents this as 00000.
- Any instruction ending with an 'I' will use an immediate value instead of a register as its last operand.

2.2.2 Arithmetic Instructions

Arithmetic instructions will add or subtract the values of Rn and Rm or Rn and a 7-bit immediate value. The result will be stored in Rd.

- ADD,Rd,Rn,Rm
  - 10000, Rm, filler5, Rn, Rd
- ADDI,Rd,Rn,imm7
  - 11000, imm7, Rn, Rd
- SUB,Rd,Rn,Rm
  - 10100, Rm, filler5, Rn, Rd
- SUBI,Rd,Rn,imm7
  - 11100, imm7, Rn, Rd

2.2.3 Memory Instructions

Memory instructions are used to access data memory. LDR/LDRI will load a 16-bit value at the address Rn + Rm or Rn + imm7 into Rt. STR/STRI will store a 16-bit value in register Rt at the memory address Rn + Rm or Rn + imm7.

- LDR,Rt,Rn,Rm
  - 10001, Rm, filler5, Rn, Rt
- LDRI,Rt,Rn,imm7
  - 11001, imm7, Rn, Rt
- STR,Rt,Rn,Rm
  - 00010, Rm, filler5, Rn, Rt
- STRI,Rt,Rn,imm7
  - 01010, imm7, Rn, Rt

3 Usage

3.1 Assembling Code

1. Make sure you have python installed and running on your computer.
2. Make sure your `.s` file and `khan.py` are in the same directory and navigate to that directory in your terminal.
3. In the command line, run `python3 khan.py <your .s file>`.
4. This will create a file called `output.txt` which you will use in the next section.

3.2 Using Khan CPU

1. Open `khan.circ` in Logisim Evolution
2. Load `output.txt` into `Instruction_Memory`
3. Run the simulation or manually run the clock until all instructions have been ran
4. Examine the `State` tab next to `Properties` and check the values of the general registers

3.3 Example Program

3.4 Code

```
ADDI,R0,R0,7
STRI,R0,R3,0
ADDI,R1,R1,21
ADD,R2,R0,R1
STRI,R2,R3,2
SUBI,R2,R2,10
LDRI,R1,R3,0
LDRI,R0,R3,2
```

3.4.1 Final Program State

- R0: 001c
- R1: 0007
- R2: 0012
- R3: 0000

4 Division of Labor

Khan was designed and implemented by Joseph DePalo. The assembler and manual were written by Rishi Bidarkota.